Query, maschere e VBA in Microsoft Access - Parte I

di Gianni Giaccaglini

SOMMARIO

(cliccare su uno dei titoli seguenti per andare al paragrafo corrispondente)

Facciamo un passo indietro 1
Prerequisiti e FAQ 2
Le Relazioni e le Query 2
Relazioni fissate tra le tabelle 2
Le query di selezione 4
Ubiquità delle query e della finestra Query 6
Tabelle e query associate a un campo: tavole di ricerca 6
Maschere, campi calcolati, codice VBA: primi cenni 8
Tabelle e query associate a controlli 9
Tabelle e query associate a una maschera 10
I campi calcolati 10
Campi calcolati in una query 10
Campi calcolati in una maschera 11

Facciamo un passo indietro

L'articolo pubblicato in precedenza su questo sito, "Introduzione al Visual Basic per applicazioni di Microsoft Access", è risultato con tutta probabilità ostico per molti. Esso infatti si rivolge a chi ha già una discreta conoscenza e familiarità col programma, cosa che non vale per la maggioranza degli utenti di Microsoft Office. Access è dotato di un'interfaccia assai gradevole e mette a disposizione molti ausili (creazioni guidate), ma i suoi innumerevoli oggetti e comandi intimidiscono. Inoltre sono in gioco delicati concetti necessari per padroneggiare un così potente DBMS (Data Base Management System). Relazionale per soprammercato. D'altronde Access è un applicativo incluso nella versione Professional di Office, un nome che la dice lunga... Ciò nondimeno non sono pochi gli utenti che migrano dalla versione normale a quella professionale o, addirittura, hanno acquistato la seconda da tempo ma senza aver avuto troppo tempo da dedicare ad Access.

Questa serie di articoli va incontro alle esigenze di chi desidera un approccio più graduale verso la meta del VBA. Un traguardo che resta comunque importante, paradossalmente molto più di quanto non capiti in altri mondi MS Office. Di macro, almeno negli usi correnti, si può anche fare a meno con Excel, per non dire di Word, mentre l'esigenza di un certo grado di automazione - a vantaggio dell'utente finale ma anche... di noi stessi - nasce ben presto se si vogliono sfruttare senza errori né ammattimenti le diverse tabelle, query, maschere, sottomaschere, report tutto il resto, di cui ben presto s'ingombra il più semplice database personale. Si pensare a una caratteristica del programma, nota a tutti ma che non guasta ribadire in apertura: Access al contrario di Word ed Excel non lavora su RAM ma direttamente su disco, per cui qualsiasi modifica fatta a un record nella vista tabellare, magari per distrazione, viene subito salvata al solo passaggio al record successivo, senza

chiederci permesso. Di qui può nascere la necessità di dotare di espliciti pulsanti di salvataggio una maschera di input, ed è solo un esempio spicciolo tra i tanti.

Prerequisiti e FAQ

L'interlocutore privilegiato di questi articoli è chiunque si trovi alle prime armi con Access, avendo però fatto già una certa pratica col suo ambiente, avendo letto qualche manuale introduttivo e conoscendo abbastanza a fondo altri programmi Office, con particolare riguardo per Excel. Una certa padronanza del Visual Basic per applicazioni è altresì richiesta (anche se in questo primo articolo del VBA di Access si avranno solo i primi assaggi). Per meglio capirci, proponiamo al lettore le seguenti **FAQ** (Frequently Asked Questions):

1) che rapporto c'è tra le relazioni fra tabelle impostate a livello database e i legami definiti nelle query?
2) è sempre indispensabile utilizzare query salvate per associarle a maschere (o report)?
3) i campi calcolati si possono avere nelle query o solo nelle maschere e nei report?

A coloro che sono certi di saper fugare questi dubbi (alcuni dei quali, forse, non troppo ben formulati, lo ammettiamo) probabilmente questi articoli non sono indispensabili. Tutti gli altri sappiano comunque che non partiremo da zero. Dando per scontata la conoscenza essenziale dell'ambiente, nonché dei concetti base dei ***database relazionali***, ci muoveremo nell'ottica dell'approfondimento e della puntualizzazione.

NOTA - Per parlare ancora più chiaro: saremo per forza incompleti se non reticenti su diversi punti, con implicito rimando a manuali specializzati.

Diciamo infine che, con questo primo articolo, non forniremo database d'esempio. Le considerazioni svolte e i casi più o meno concreti discussi all'inizio faranno, ove occorra, riferimento al database **Gestione contatti**, che si può ottenere con una facile creazione guidata offerta da Access. In fondo all'articolo abbiamo invece suggerito di riferirsi al database **Clienti01** pubblicato con l'articolo precedente ("Access VBA - introduzione").

Le Relazioni e le Query

Cerchiamo innanzitutto di riassumere le (principali) caratteristiche di Access, con lo scopo di commentarle più che di descriverle in dettaglio, la qual cosa esula dagli scopi che ci siamo prefissi e dallo spazio-tempo disponibile.

Si tratta di un DBMS rigorosamente relazionale che si differenzia da altri suoi simili per il fatto di incorporare in un unico file di estensione MDB tutti i suoi "oggetti" o componenti che dir si voglia, a partire da quelli base, le **tabelle**, mentre altrove queste sono file separati. Naturalmente stiamo parlando dei database tipici di Access, che in realtà è in grado di trattare anche database più o meno remoti e diversamente organizzati. Noi qui ci limitiamo ai database locali, magari formati da un numero di tabelle limitato, ma sufficiente per implementare un'applicazione gestionale più o meno complessa.

Relazioni fissate tra le tabelle

Per capire questo punto invitiamo a far creare da Access il più semplice "modello" offerto dal programma, con **File ⇒ Nuovo**, scegliendo poi **Gestione contatti**. Con pochi semplici passaggi ci viene creata un'intera struttura di tabelle, maschere e report bell'e pronti, che dobbiamo solo riempire coi nostri dati.

Il lettore è caldamente invitato a sperimentare fin d'ora di persona questo "canovaccio", navigando fra i suoi componenti e inserendo al più presto dati opportuni nelle varie tabelle, con o senza ricorrere alle maschere.

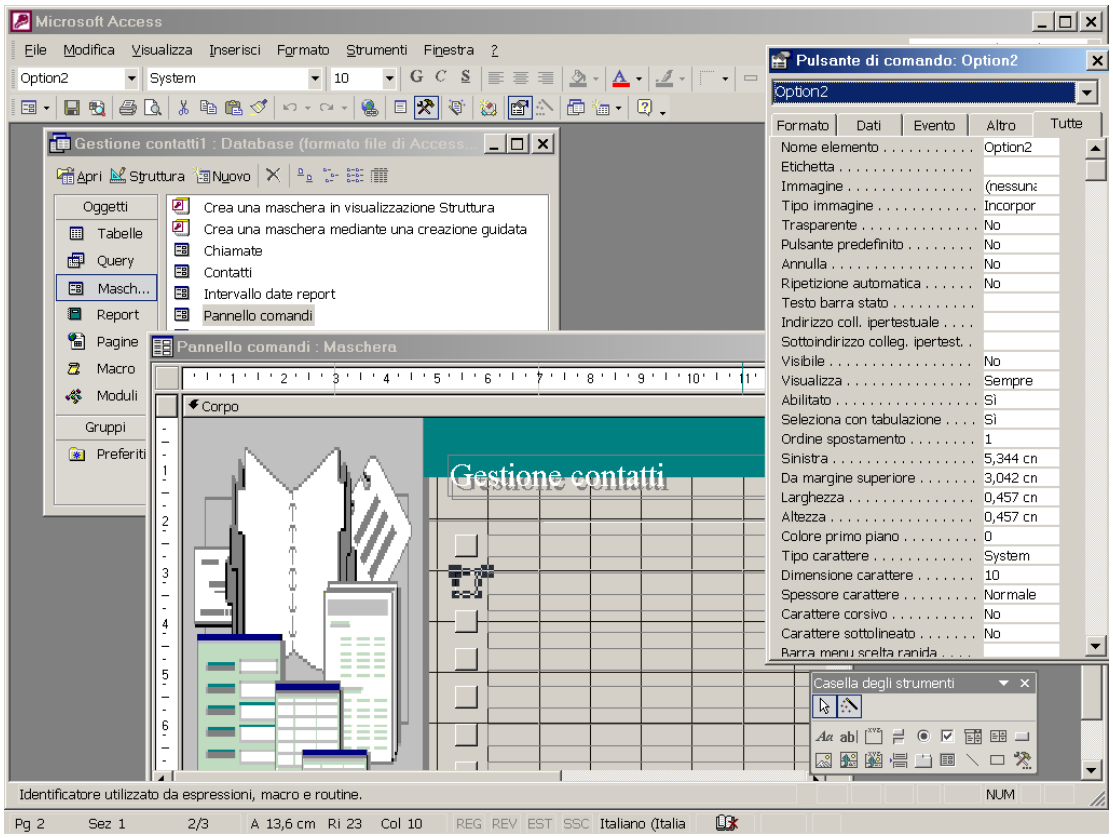
NOTA - Si consiglia, in particolare, di riempire direttamente la tabella Chiamate, inserendo il campo DataChiamata con valori per lo più precedenti a quelli della data odierna (diciamo di un paio di mesi). La maschera invece offre *=Date()*, sia pure come default non obbligatorio.

La cosa che più salta all'occhio è la presenza di una maschera d'apertura dotata di pulsanti di tipo Interruttore (mutuamente esclusivi, grazie a opportuno codice VBA), ciascuno dei quali attiva un compito indicato a chiare note. Ad esempio "Immetti/Visualizza Contatti" apre a sua volta una maschera per; Visualizza report in anteprima e altre.

Questa particolarità è semplice da descrivere, per cui liquidiamola subito, anche perché ci dà il destro per enfatizzare non solo la natura (almeno auspicabilmente) applicativa dei database Access ma anche l'importanza delle ***maschere*** nel suo mondo.

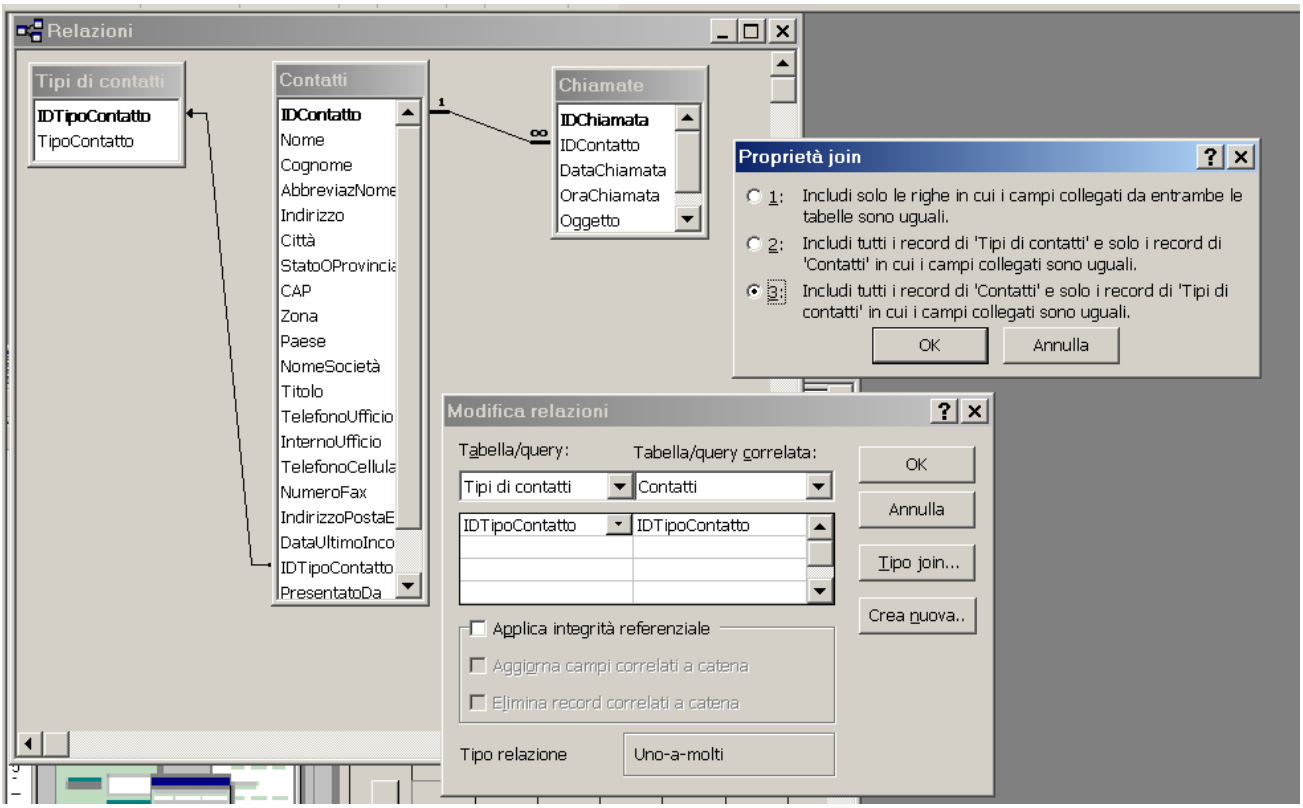
Se si attiva la finestra Database (*) si scopre subito che la maschera d'apertura ha nome "Pannello di comando" e attivando l'icona **Struttura** (riga+squadra+matita) e quella delle **Proprietà** si possono studiarne componenti e attributi.

NOTA (*) - Questa e altre manovre basilari sono date per arcinote, per cui non le descriveremo che per sommi capi.



Circa l'auto-avvio, chi proviene da Excel o Word sicuramente pensa all'intervento di una routine d'evento del tipo *Worksheet_Open*, ma NON è così: Access è straricco di eventi, ma sono tutti relativi a maschere e report, mancano completamente eventi di apertura, chiusura e simili gestibili in VBA. Ma allora come funziona l'apertura automatica? Semplice, è una proprietà del database che si fissa con **Strumenti ⇒ Avvio...**, indicando poi, nella finestra susseguente, la maschera da far apparire all'avvio: la maschera **Pannello di comando**, nel nostro caso.

Proseguendo in questo rapido giro esplorativo, esaminiamo le relazioni di **Gestione contatti1.mdb** (ci si perdoni se per distrazione abbiamo accettato il nome default) scegliendo **Strumenti ⇒ Relazioni...** o l'equivalente icona. Con un clic destro sulla linea che lega i campi IDTipoContatto delle due tabelle di sinistra evidenziamo la natura della prima relazione, poi il pulsante **Tipo join** ci svela un altro altarino:



La precedente figura evidenzia una relazione Uno-a-molti tra le due tabelle di sinistra e diamo per noto il suo significato, ricordando solo che nella tabella Contatti (lato “Molti”) vi sono più codici IDTipoContatto mentre nella tabella del lato “uno” l’omonimo campo identifica una chiave unica, cui corrisponde un solo TipoContatto (putacaso: “Familiare”, “Ufficio”, “Amici” e simili). Lasciamo per esercizio l’analoga mossa del clic destro sul secondo legame con cui si evidenziano le caratteristiche della relazione tra le tabelle Contatti e Chiamate e puntualizziamo quanto segue.

- **Integrità referenziale**. Si può applicare o meno, e, se la si impone, la si può definire con una, due o nessuna fra queste modalità: *Aggiorna campi correlati a catena* e *Elimina record correlati a catena*;
- **tipo di join**: nelle operazioni di query, si tiene conto o solo delle righe (ossia record) i cui campi correlati sono uguali (opzione 1, default, della proprietà join, v. figura precedente) oppure (opzioni 2 e 3) prendere comunque tutti i record di una delle due tabelle.

L’integrità referenziale, fattore di assoluta rilevanza nei database relazionali, mira a evitare che vi siano record orfani nelle tabelle correlate. Lasciando al lettore il compito di meditare sulla relazione presente, nel modello Gestione contatti, tra Contatti e Chiamate, preferiamo fare il classico esempio di una tabella Articoli correlata a una tabella di Spedizioni, tramite un CodArticolo: se si elimina il record di un articolo, tutti i record delle spedizioni specifiche rimangono privi di altre notizie del defunto articolo (prezzo, giacenza ecc.).

Il tipo di join è anch’esso importante, perché se applicato in modo scorretto potrebbe dar luogo a risultati indesiderati. Nel ribadire che su questa materia il rinvio a manuali è d’obbligo, per farci capire consideriamo stavolta il caso del modello. Se si applicasse il tipo 1 di join, detta **equi-join**, la query produrrebbe solo i record che hanno un TipoContatto codificato nella tabella Tipi di contatti, per cui verrebbero esclusi i soggetti che non abbiamo deciso se catalogare come “Familiare”, “VIP” eccetera. Con la join di tipo 3 questa drastica esclusione viene evitata.

Le query di selezione

L’integrità referenziale viene segnata con bordo spesso ai due lati della linea di correlazione, con in più l’indicazione **1**, sul lato Uno, e ∞ (infinito), sul lato Molti. Non c’è

nessun obbligo a farlo, ma è generalmente molto importante che relazioni di questo tipo siano correttamente stabilite a livello Database. Questo anche grazie a una virtù di Access che vale la pena di evidenziare:

Se si fissano relazioni fra le tabelle nel progetto del database il programma impedisce, automaticamente, che vengano violati i criteri di integrità referenziale, in qualunque operazione di aggiornamento.

In altri termini, l'integrità referenziale impedisce che eliminando un record sul lato Uno si creino dei record "orfani" sul lato Molti. E questo, si badi bene, anche nelle normali operazioni di tipo manuale, su un foglio dati o una maschera.

NOTA - In questa trattazione, tutt'altro che sistematica, non si parla della relazione più semplice, la relazione Uno-a-Uno, che viene data per nota.

Passiamo ora alle operazioni di query, limitandoci per ora a quelle *di selezione*, ossia volta ad estrarre da una o più tabelle (correlate) i dati che interessano. Qui s'innesta a proposito la FAQ numero 1, che ripetiamo:

che rapporto c'è tra le relazioni fra tabelle impostate a livello database e i legami definiti nelle query?

Non perdiamo tempo e diamo subito la risposta, che si articola in due punti:

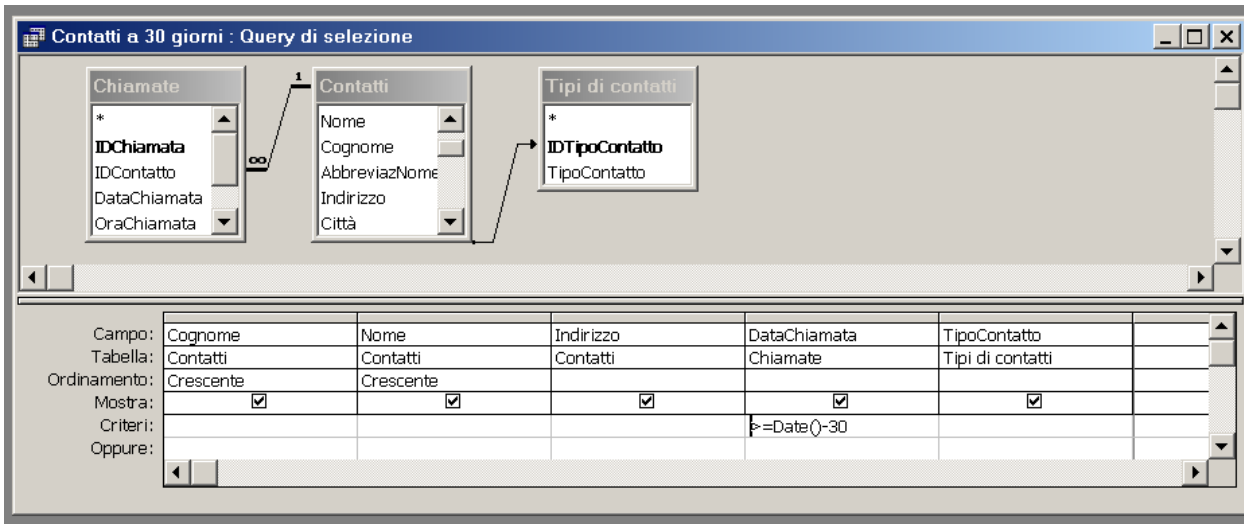
1) *se si va a definire una query su tabelle tra le quali sono impostate relazioni a livello database tali legami vengono ereditati dalla query, per default;*
2) *nulla però vieta che tali relazioni vengano modificate o impostate diversamente, nel qual caso sono valide soltanto nell'ambito di quella query.*

Nelle query si possono fissare tutte le relazioni che ci piacciono, beninteso se hanno senso e ne vale la pena, ma quelle fissate nel progetto restano in piedi e, quel che più conta, restano comunque valide le regole di integrità referenziale fissate.

Se si riflette, questo ruolo censorio viene esercitato con le query di aggiornamento (con particolare riguardo per quelle che cancellano record!). Ma noi, in questa fase, ci occuperemo delle query più comuni, ossia le ***query di selezione***. Ne mostriamo subito una all'opera, anche per dar corpo alla discussione qui in atto. Sempre con riferimento al database Gestione contatti1 si compiano questi passi, che i più dovrebbero conoscere:

1. nella finestra Database, scegliere il pulsante **Query**;
2. sul pannello di destra della stessa finestra, scegliere **Crea una query in visualizzazione struttura** (la seconda opzione, facilitata da una **creazione guidata**, è lasciata per esercizio;
3. compare la finestra ***Query*** accompagnata dalla finestra ***Mostra tabella***, attiva, con l'elenco delle tabelle (o magari delle query già salvate, per l'operazione di sub-query), dalla quale è facile attingere e portare le tabelle che interessano nella finestra ***Query***; Selezioniamole tutte le tre del modello.
4. si chiude ***Mostra tabella*** poi nella finestra ***Query*** si trascinano i campi voluti dal pannello superiore a quello inferiore, e impostare gli "esempi" ossia i criteri desiderati.

Si avrà una situazione del tipo seguente, il cui scopo è selezionare i "contatti" (i nostri interlocutori) con cui ci siamo parlati negli ultimi 30 giorni:



La cosa che ci preme sottolineare in questo contesto è che le relazioni impostate a livello database sono ereditate dalla nuova query, in prima battuta. Ma nulla ci vieta di definire una query caratterizzata da relazioni diverse e/o con tipi di join differenti. Per esempio, dopo aver tolto il criterio sulla DataChiamata, potremmo, nella relazione di sinistra, modificare il join da tipo 1 (equi-join) a tipo 2 (right-join). In tal modo la selezione fornirà anche i soggetti con cui non ci siamo parlati (meglio: non hanno riscontri nella tabella Chiamate). Questi hanno il campo DataChiamata vuoto, mentre con la join di tipo 1 vengono esclusi. Comunque, anche se salviamo la nostra query "anomala" le relazioni a livello database restano immutate.

Ubiquità delle query e della finestra Query

Come è noto, almeno per sentito dire, Access come tutti i relazionali doc si basa sull'SQL (Structured Query Language), un linguaggio di comandi che agiscono sul database secondo una sintassi standard. Eccone un esempio:

```
SELECT Contatti.Cognome, Contatti.Nome, Contatti.Indirizzo, Chiamate.DataChiamata,
[Tipi di contatti].TipoContatto FROM ([Tipi di contatti] RIGHT JOIN Contatti ON
[Tipi di contatti].IDTipoContatto = Contatti.IDTipoContatto) INNER JOIN Chiamate ON
Contatti.IDContatto = Chiamate.IDContatto
WHERE (((Chiamate.DataChiamata)>Date()-30))
ORDER BY Contatti.Cognome, Contatti.Nome;
```

È una direttiva abbastanza complicata, comunque i verbi base evidenziati in maiuscolo denotano un'operazione di selezione (SELECT), secondo un criterio (WHERE), un certo INNER JOIN e un tipo di ordinamento (ORDER BY). E non ci vuol molto a scoprire che la clausola rispecchia esattamente la query impostata nella figura precedente: è sufficiente, stando nella ***finestra Query***, scegliere **Visualizza ⇒ Visualizza SQL** per ottenerla (e, in certi casi, copiarla altrove se fa comodo). Orbene Access ci evita l'SQL in tutte o quasi le circostanze in cui ci occorre, mascherandolo con una finestra Query nella quale è facile impostarne i parametri secondo il modello QBE (Query By Example).

Ci siamo permessi di parlare, nel titolo di questo paragrafo, di "ubiquità" delle query e della relativa finestra per due motivi:

- le query o, meglio, l'SQL sono in gioco non soltanto come "oggetti" (salvati o meno) ma anche come clausole SQL affibbiate a maschere e a controlli;
- anche in questi casi entra in gioco la finestra Query, automaticamente.

Tabelle e query associate a un campo: tavole di ricerca

In Access il controllo più frequente è la *casella di testo*, il più delle volte destinata ad accogliere il dato di un campo. Sono considerate caselle di testo anche quelle normali di una tabella o di una query, insomma in un foglio dati. Un caso semplice ma emblematico è dato dalle tavole di ***ricerca*** (lookup) che si possono impostare in certi campi per limitare a

una tabella (o query!) di voci prestabilite le scelte dell'utente. Ciò può avvenire in una volgare tabella, come pure in un controllo di una maschera. Per fissare le idee descriviamo la tabella di ricerca già stabilita nel campo IDTipoContatto della tabella Contatti:

1. aprire Clienti nella visualizzazione Struttura (clic sull'icona riga+squadra+matita);
2. attivare il campo IDTipoContatto nel pannello superiore della finestra della Struttura;
3. passare al pannello inferiore (Proprietà campo) e attivare la casella *Origine riga*, accanto alla quale spuntano una freccia in basso è l'icona "Generatori", a tre puntini (...): un clic sulla prima apre l'elenco di altre tabelle e query (salvate);
4. attivando invece i Generatori si vedrà apparire una finestra Query (*) in cui si vedono (o, magari, modificano) i parametri della query associata al campo IDTipoContatto.

NOTA (*) - In questi casi la finestra prende il nome di *Generatore di query*, ma è sempre lo stesso strumento...

Il risultato, come si sa già o constata, è che l'utente può scegliere solo tra i termini presenti nella tabella [Tipi di contatti], come si può anche desumere dalla corrispondente clausola SQL:

```
SELECT DISTINCTROW [Tipi di contatti].*
FROM [Tipi di contatti] ORDER BY [Tipi di contatti].TipoContatto;
```

Già che ci siamo, precisiamo il significato di DISTINCTROW (*): la sua presenza fa sì che, in caso di doppioni ne venga preso uno soltanto. Ricordiamo poi che per imporre tale restrizione - pleonastica nel caso specifico, ma indispensabile quando si attinge a tabelle in cui abbondano ripetizioni (in quanto non rivolte a soli scopi di lookup) - nella finestra Query si deve attivare l'icona Proprietà (o dare un clic destro) e impostare a Si la casella *Record univoci* (*).

NOTA (*) - Questa restrizione e il relativo termine sono peculiari di Excel, nell'SQL standard la sola chiave è DISTINCT, che corrisponde a *Valori univoci*, nome della casella appena sopra alla *Record univoci* della finestra Proprietà.

Vediamo ora, sempre in qualche dettaglio, il procedimento di creazione di una query per tabella di lookup. Ci riferiremo, stavolta, al database **Clienti.mdb** presentato nell'articolo precedente ("Access VBA - introduzione"): aprendo la tabella Pagamenti si constata subito che nel campo CodCliente (intestato "Abbonato", ma il nome vero del campo è CodCliente) compaiono voci quali "Rossini Mario", "Corneli Camillo" ecc., frutto di una query che aggiunge al CodCliente un ***campo calcolato*** ottenuto concatenando con l'operatore **&** i campi Cognome e Nome della tabella Clienti. Il lettore si sforzi di capire il marchingegno, sulla falsariga di quanto testé descritto.

Per ripercorrere invece il lavoro creativo, si seguano i passi seguenti, esattamente se si vogliono evitare frustrazioni e risultati strani e inattesi.

1. Eliminare la relazione tra le due tabelle (prendendo nota dei suoi parametri, per il successivo ripristino!). Ciò è necessario, altrimenti Excel rifiuta di ricreare una tabella di ricerca perché questa implica tale relazione (come si vedrà).
2. Nella struttura di Pagamenti, in corrispondenza di CodCliente cancellare la clausola SQL della casella *Origine riga*, poi in *Visualizza Controllo* scegliere "Casella di testo". Queste mosse eliminano del tutto l'origine e la forma del lookup.
3. Tornando al pannello superiore, nella cella accanto al nome del campo CodCliente, aprire la casella a discesa e scegliere non un tipo dati, bensì **Ricerca guidata...**, avviando così un wizard dedicato.
4. Nei primi 3 passaggi scegliere, nell'ordine: "Ricerca valori in una tabella o query..."; la tabella Clienti; i campi di Clienti CodCliente e Cognome (trascinarli dal riquadro sinistro al destro col pulsante **>**).
5. Al quarto passaggio assicurarsi che la casellina *Nascondi colonna chiave* sia attiva, poi premere **Fine** all'ultimo passaggio.

Salvando la struttura e visualizzando la tabella Pagamenti si constaterà che nella colonna Abbonato (CodCliente, in realtà) si vedono i vari Cognomi, mentre il dato "vero"

sottostante" è un codice. Non è finita: aprendo la finestra delle Relazioni si vedrà che la connessione Uno-a-molti, tramite il CodCliente delle due tabelle è stata ripristinata. Ci restano due compiti:

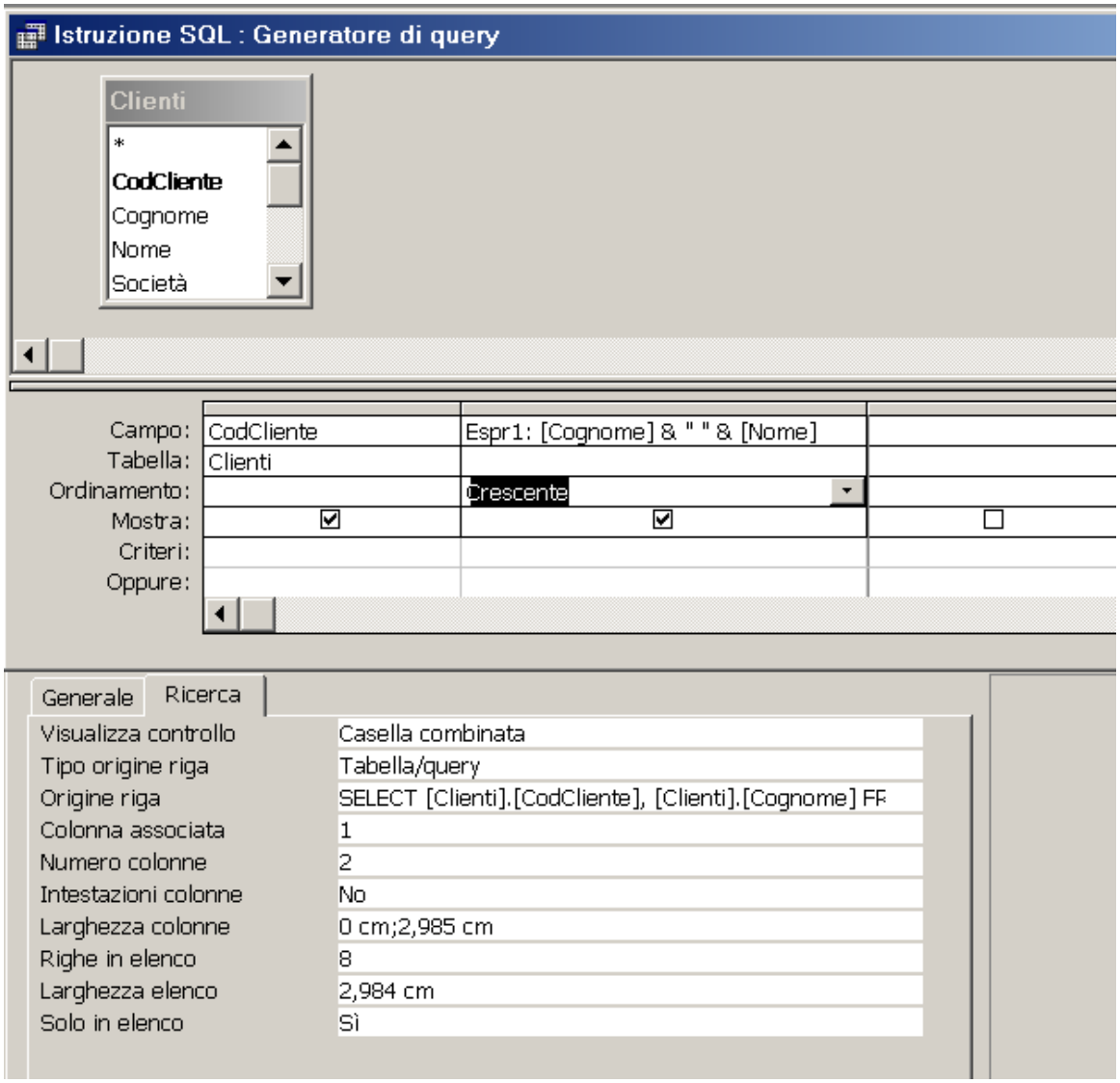
6. assegnare alla risuscitate relazione l'integrità referenziale e il tipo di join originari;
7. modificare la query per concatenare Cognome e Nome.

L'ultima operazione parte cliccando sui tre puntini accanto alla casella *Origine riga* poi nel Generatore di query digitare nel campo Cognome, brutalmente: Cognome & " " & Nome per premere **Invio** e fissare l'ordinamento crescente. Il campo calcolato diventa:

espr1: [Cognome] & " " & [Nome]

Osservazioni volanti: a) *espr1* è il nome default (modificabile) assegnato a un campo calcolato di una query; b) i nomi di campo per regola vanno racchiusi tra parentesi quadre che si possono omettere solo se tali nomi non presentano spazi (nel qual caso Excel aggiunge le quadre per bontà sua).

La figura che segue riassume in qualche modo il risultato:



I valori *Colonna associata* = 1 e *Numero colonne* = 2 significano che la relazione si basa sul primo campo mentre la query restituisce 2 colonne, di cui è visibile solo la seconda, calcolata.

L'analisi del codice SQL risultante è lasciata per esercizio.

Maschere, campi calcolati, codice VBA: primi cenni

Parlando di tabelle di ricerca in una tabella ci siamo dilungati forse anche troppo sui dettagli operativi, ma le questioni e... gli imbarazzi in cui ci si imbatte in siffatti contesti meritavano un certo spreco si spazio. Possiamo però dire, sempre a proposito di ubiquità delle query, che lo stesso procedimento, con qualche variante, si può applicare nelle maschere e nei report (vedremo). Per quanto riguarda le prime la cosa può riguardare:

- l'intera maschera
- singoli controlli, del tipo casella combinata e casella di riepilogo.

Questo articolo preliminare, ripetiamo, mira più a una discussione sui principi che a una descrizione sistematica. Prossimamente descriveremo le maschere in modo più approfondito, dando per il momento come risapute le nozioni base sulle maschere e sui controlli in esse racchiusi. Detto en passant, solo abbastanza più avanti ci ripromettiamo di parlare dei report, che pure molti manuali descrivono fin dall'inizio.

Una cosa importante va però anzitutto fatta (o ribadita, chi già la sapesse), a scanso di equivoci. Le maschere (in inglese ***Form***) possono essere di due tipi:

- *associate*
- *non associate*.

Una maschera del secondo tipo in sostanza è una finestra di dialogo, parente stretta delle UserForm di Excel e "non associata" significa che non ha legami con insiemi di record (recordset) provenienti da tabelle o query.

NOTA - A dir il vero non è escluso che una maschera Access "non associata" non includa controlli collegati a dati, ma per ora non sottilizziamo su situazioni ibride...

Una maschera associata, la più peculiare e frequente nel mondo Access, è invece destinato all'input ed, eventualmente, modifica di dati. In questa sede si farà riferimento al più comune tipo di maschera associata, il tipo a colonne, composto da caselle che corrispondono a campi di un recordset.

Per continuità con il tema del paragrafo precedente - "Tabelle e query associate a un campo: tavole di ricerca" - cominceremo coi controlli di una maschera associati a tabelle o query.

Tabelle e query associate a controlli

Descriviamo le prime mosse della strategia più interessante, quella di creazione guidata (wizard).

NOTA - Il ricco ambiente Access in molti casi offre sia un wizard sia un procedimento manuale. Il secondo, indispensabile in sede di modifica, richiede di agire sulle *proprietà*. La cui finestra si apre, come tutti dovrebbero sapere mediante l'icona a forma di manina che punta l'indice su un foglietto.

1. Aperta la nostra maschera nella vista Struttura, assicurarsi che sia aperta la Casella degli strumenti e, in questa, attivare il secondo pulsante (dopo il puntatore a freccia), "Creazioni guidate controllo".
2. Selezionare il controllo **Casella combinata** (o, se si preferisce, Casella di riepilogo), trascinarlo e dimensionarlo sulla maschera more solito: appare il primo passaggio della creazione.
3. Scegliere "Ricerca valori in una tabella o query...".
4. Proseguire con le stesse mosse viste per le tavole di ricerca del campo di una volgare tabella!

Infatti chiunque non abbia memoria labile si rende conto che questo wizard è lo stesso e che, in parole povere, le tavole di ricerca si possono avere sia in una tabella che in una maschera. C'è solo una (piccola) differenza da rimarcare:

5. nella finestra Proprietà del controllo appena creato, nella scheda ***Dati***, si constata che la proprietà *Tipo riga* contiene "Tabella/query", mentre *Origine riga* riporta una frase SQL del tipo "SELECT ... FROM ...", ma è la stessa cosa di cui sopra e, come sopra, un clic sull'icona a tre puntini ci apre la finestra Query (generatore).

NOTA - I più esperti imparano, prima o poi, se non a digitare a mano frasi SQL, almeno a copiarle negli Appunti per incollarle in altri controlli...

Tabelle e query associate a una maschera

L'associazione di una maschera comune a una tabella dà luogo, semplicemente, all'indicazione del nome di tale tabella nella proprietà *Origine record* nella scheda ***Dati*** della finestra delle Proprietà. Più interessante è l'associazione a una query, vediamo dunque come si può ottenere in una caso semplice.

1. Si crei una query basata su una sola tabella, diciamo la solita Clienti, in cui nella casella Criteri del solo campo Città venga scritto "Milano". Nessuno si sorprende nel constatare che la neonata query seleziona solo i clienti milanesi.
2. Salvare la query col nome **Clienti Milano**.
3. Creare una nuova maschera chiamandola anch'essa **Clienti Milano**, con creazione guidata o meno, indicando come *Origine record* la query Clienti Milano.

Anche qui nessuna sorpresa, anche la maschera Clienti Milano naviga solo fra i meneghini. Ma poniamoci ora la FAQ numero 2: è sempre indispensabile utilizzare query salvate per associarle a maschere (o report)? La risposta di chi ci ha seguito non può essere che negativa, vediamo solo di sperimentarla.

4. Eliminare la query Clienti Milano o, meglio ancora, rinominarla **Clienti Milano salvata**: in entrambi i casi la nostra maschera fallisce.
5. Apriamola in vista Struttura e cancelliamo "Clienti Milano" nella casella *Origine record*.
6. Attiviamo l'icona generatori e, nella finestra Query, ripetiamo i passaggi compiuti al passo 1.

Non ci vorrà molto a constatare che il risultato è lo stesso di prima, salvo che ora *Origine record* reca la frase SQL del tipo seguente:

```
SELECT Clienti.CodCliente, Clienti.Nome, Clienti.Società, Clienti.Città, Clienti.Scadenza
FROM Clienti
```

```
WHERE (((Clienti.Città)="Milano"));
```

Un'alternativa per esperti (esperti per modo di dire: tutti dovrebbero impratichirsi con queste piccole astuzie) consiste nel copiare la frase SQL della query salvata, nella finestra fatta apparire con **Visualizza ⇒ Visualizza SQL**, per poi incollarla nella casella *Origine record.*

Il discorso sulle maschere associate a query non termina qui. In questo articolo ci siamo limitati a query basate su una singola tabella, ma (come i più bravi & audaci possono già sperimentare) con le query caratterizzate da join fra più tabelle le cose si complicano. Lo vedremo nel prossimo articolo parlando di maschere e sottomaschere, fra loro collegate.

I campi calcolati

I campi calcolati o, per l'esattezza, le celle contenenti formule che puntano ad altre celle sono la merce più abbondante in Excel. Chi proviene da questo ambiente potrebbe stupirsi che i campi calcolati nelle tabelle sono assenti. Anzi, sono tabù, in ossequio a una precisa e tassativa norma dei database relazionali, pertanto ne riportiamo la motivazione, senza discuterla: se un dato deriva da altri dati è inutile appesantire la memoria di massa, tanto lo si può calcolare al momento in cui occorre.

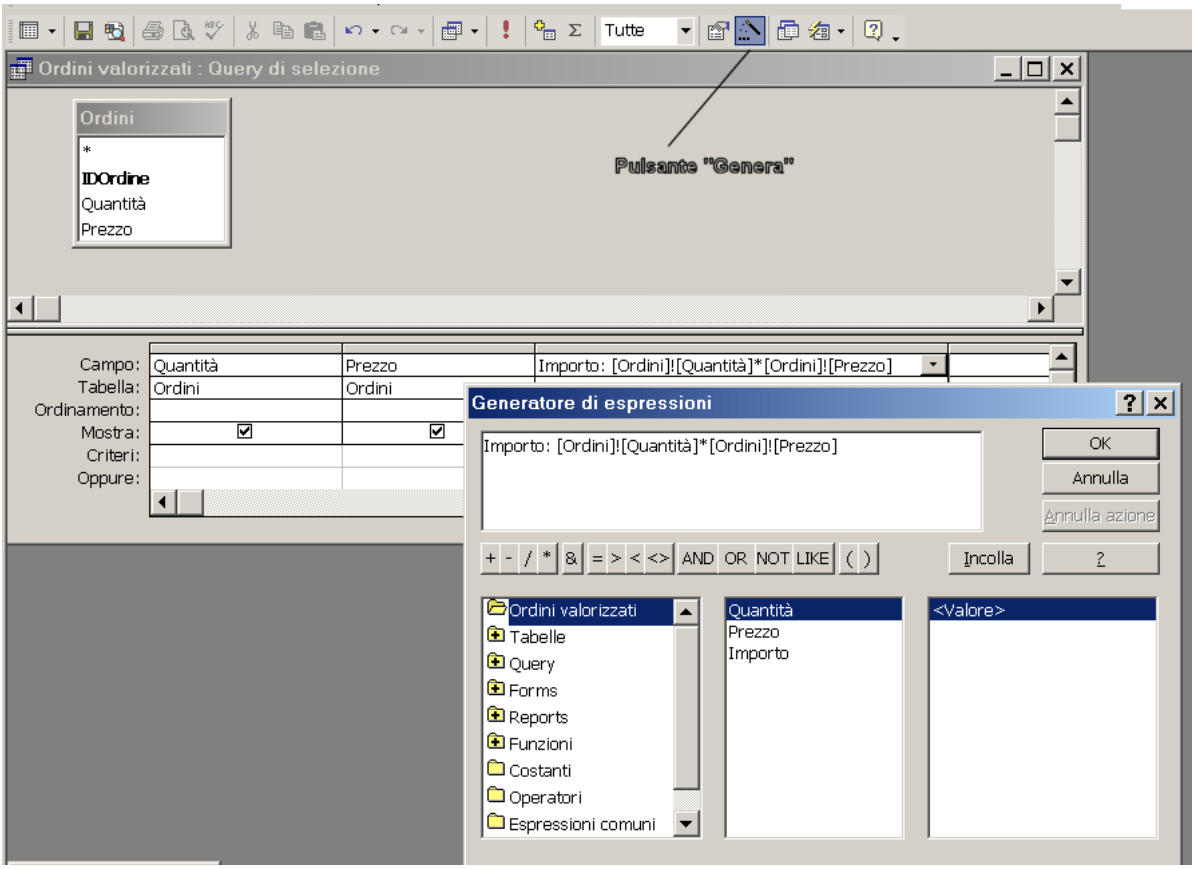
Campi calcolati in una query

I sacri testi di solito affermano che i campi calcolati si possono avere con le maschere e i report, per cui qualcuno potrebbe porsi la FAQ numero 3. Alla quale, però, abbiamo già dato una implicita quanto concreta risposta affermativa nel descrivere la tabella di ricerca che fornisce il concatenamento *Cognome & " " & Nome*, tramite, giustappunto, un campo calcolato. Questo, se si ricorda, assume il nome default *espr1* (successivamente: *espr2, espr3,...*), che possiamo rendere più pregnante se ci piace.

Nel caso citato si trattava di una query ad uso di una tavola di ricerca, vediamo ora una variante, più propriamente aritmetica e inerente una query salvata. Per fissare le idee, proponiamo il classico caso di una tabella Ordini contenente i campi Quantità e Prezzo, la cui creazione è un (facile) compito affidato al lettore, che poi non dovrebbe avere difficoltà a creare una query basata su Ordini, comprendete i campi detti. Dopodiché, nella terza colonna del pannello inferiore della finestra Query, è altrettanto agevole ottenere la non sorprendente espressione, con l'ausilio del Generatore di espressioni (icona bacchetta magica). Si parte con un clic su tale icona poi si pescano nella finestra del Generatore i termini, gli operatori e le funzioni che occorrono.

NOTA - La pratica con questo Generatore, utilizzabile in diversi contesti del mondo Access, è lasciata al paziente lettore, che saprà divertirsi da solo.

La figura che segue illustra il risultato ottenuto, relativo alla query già prodotta e salvata come **Ordini valorizzati**, ma le mosse da compiersi nella creazione sono pressoché identiche.



Al posto di espr1 è stato scritto Importo, che diventa il nome del terzo campo, calcolato, della nostra query. Lanciarla (premendo il pulsante punto esclamativo, chi ancora non lo sapesse).

NOTA - I termini [Ordini]![Quantità] e [Ordini]![Prezzo] potevano essere ridotti a [Quantità] e [Prezzo], visto che è in gioco una sola tabella. La sintassi col ! (che come molti dovrebbero già conoscere) verrà ridiscussa a suo tempo.

Per la cronaca, ecco la direttiva SQL che agisce in sottofondo nella nostra query calcolata:

```
SELECT Ordini.Quantità, Ordini.Prezzo,[Quantità]*[Prezzo] AS Importo
FROM Ordini;
```

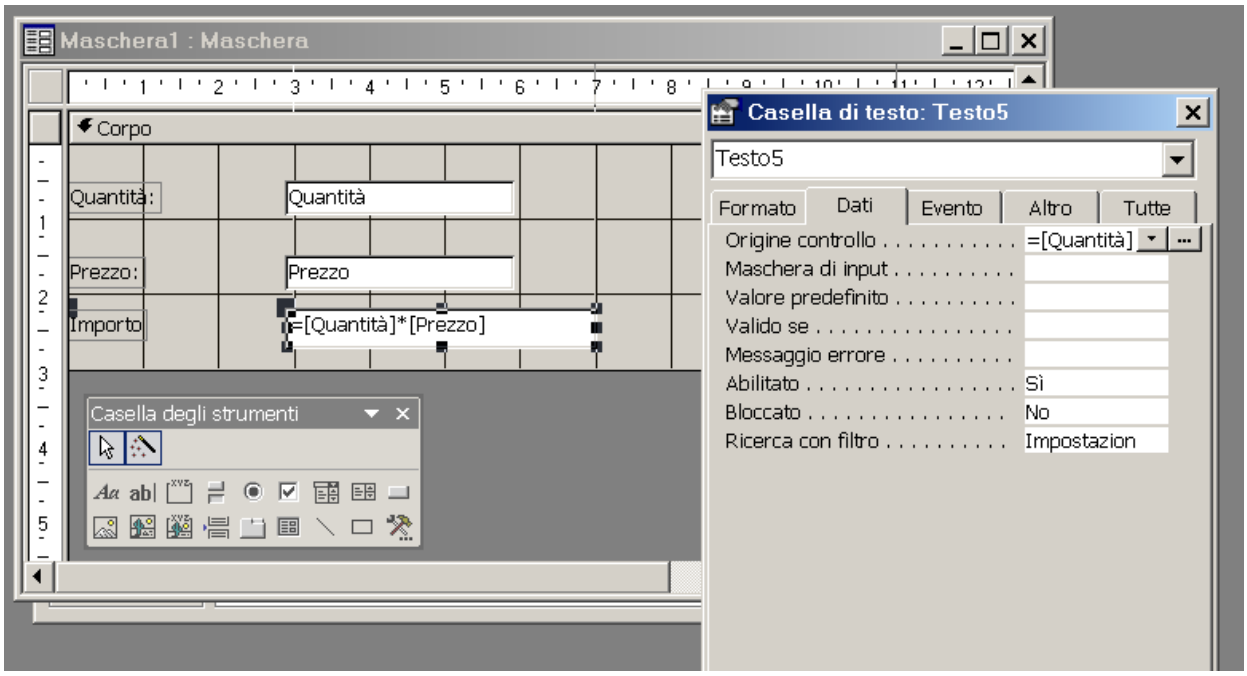
Come ripetiamo, la si può visualizzare col comando **Visualizza** ⇒ **Visualizza SQL** della finestra Query.

Il concetto di *dynaset*. Il discorso sui campi calcolati in una query ci sembra l'occasione giusta per accennare a questo concetto, che sarà ripreso a tempo e luogo. Le query di selezione, a differenza delle tabelle, che fanno parte del database in permanenza, esistono solo all'atto del loro lancio, per questo si parla di **dynaset**, insiemi di dati dinamici, che impegnano la memoria di massa solo temporaneamente. Ecco spiegato perché i dynaset (o loro parenti, come gli snapshot) possono violare le regole delle tabelle "normalizzate", sia per quanto riguarda le ridondanze sia per quanto riguarda, per l'appunto, i campi calcolati.

Campi calcolati in una maschera

Per avere un campo calcolato in una maschera occorrono due cose: una casella di testo aggiuntiva rispetto a quelle relative a campi e una formula come origine di tale controllo. Tale formula va inserita nella casella della finestra Proprietà del controllo denominata, per l'appunto, *Origine controllo*.

Chi è giunto a questo punto saprà cavarsela da solo, con l'aiuto di wizard o meno a creare una maschera del genere per la tabella Ordini. La figura che segue illustra la situazione a un passo dalla fine:



Si è partiti con la Creazione maschera in visualizzazione Struttura e l'azione appena precedente, vista la banalità della formula, è stato non il ricorso al Generatore, ma la brutale digitazione di "=Quantità*Prezzo". Le parentesi quadre le aggiunge, bontà sua, Access.

Che altro dire? Soltanto due constatazioni alquanto ovvie:

a) la formula in un controllo inizia con "=" e sottintende a primo membro il contenuto del controllo (come in Excel); si rifletta sulla pur lieve differenza con la sintassi appena vista dei campi calcolati delle query, ove il primo membro va indicato (per forza! è il nome del campo aggiuntivo del dynaset);

b) lanciando la maschera si vede che il campo Importo non è modificabile.

Facciamo infine un'osservazione semplice ma concettualmente significativa:

dal momento che una maschera può essere associata a una tabella oppure a una query ne deriva che un'alternativa possibile consiste nel creare la precedente maschera associandola alla query salvata Ordini valorizzati.

Né è da escludere la possibilità che tale query, anziché salvata, sia esclusiva della maschera stessa. È un esperimento, forse eccessivo, che lasciamo ai più curiosi e spericolati.

Il Visual Basic di Access: preliminari con un piccolo antipasto

Il mondo di Access è popolato di una fauna composta da molte specie o, meglio, oggetti. All'impianto di fondo delle tabelle e delle relazioni fissate fin dal progetto di un certo database, si uniscono query, maschere, report per citare i più importanti). Questi componenti possono essere collegati e arricchiti in vari modi dai seguenti strumenti, che dovranno costituire un cocktail organico in un progetto ben strutturato:

- ***clausole SQL***
- ***dati calcolati***
- ***routine Visual Basic*** (per applicazioni, ***VBA***).

Rimangono in vita, per compatibilità verso il basso, le ***macro*** tipiche di Access. Nate con la versione 2.0, esprimono tutte le possibili azioni che si possono compiere su tabelle, query, maschere e report, ma questi arnesi sono considerati obsoleti da Microsoft, fin dalla versione 97 (e con Access 2000 e 2002 se carichiamo un vecchio database munito di macro

“all’antica” veniamo invitati a farcele tradurre in VBA, cosa che il programma compie in un fiat e senza errori, se aderiamo alla proposta.

Spunti generali

Non guasta sapere che le vecchie macro sono nella quasi totalità fedelmente rispecchiate, nella nuova sintassi VBA, dai metodi e dagli argomenti dell’oggetto ***DoCmd***. Questa parola chiave sintetizza *Do Command,* esegui comando, sicché per fare un solo esempio, *DoCmd.OpenForm “MascheraMiaBella”...* non fa che tradurre in VBA il vecchio comando macro di apertura di una maschera dal patetico nomignolo.

Ovviamente il VBA di Access fa molto di più di quanto non era possibile con le macro tradizionali, ma questo lo vedremo in seguito.

Siamo alla frutta, pertanto ci limitiamo a un piccolo esempio, a titolo di assaggio (e contentino per chi sperava in più ampi ragguagli in merito). Così facendo, oltre a completare in qualche modo la panoramica su Access, i neofiti potranno iniziare a familiarizzarsi con le particolarità del Visual Basic di Access.

NOTA - Un prerequisito rimane la conoscenza del VBA di altri applicativi, come quello di Excel, e in particolare l’utilizzo dell’Editor Visual Basic. Che, salvo eccezioni, non verrà descritto.

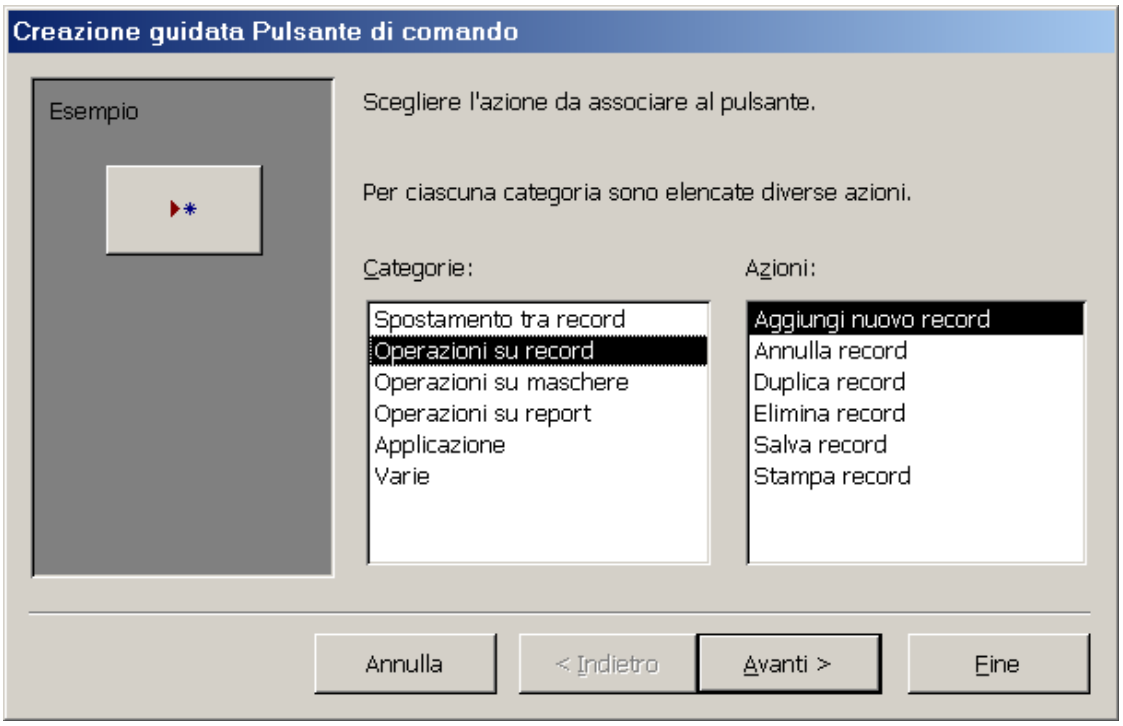
Terminiamo queste note generali con una particolarità del VBA di Access. All’Editor Visual Basic si accede in vari modi (che vedremo di volta in volta), qui diciamo che oltre che col classico comando S**trumenti _⇒ Macro ⇒ Visual Basic Editor** o l’altrettanto classica scorciatoia **Alt**+**F11** Access mette a disposizione una particolare icona, etichettata “Codice” (sottinteso: VBA) nella barra degli strumenti. Facile da trovare, comunque si sappia che è lo stesso simbolo di un modulo di classe.

Creazione di un pulsante e relativa routine d’evento

Entriamo in medias res con un esempio spicciolo, descritto passo passo. Stavolta useremo il database Gestione contatti, proponendoci di aggiungere un paio di pulsanti alla maschera Contatti. Il primo di questi, che esamineremo più da presso, servirà semplicemente all’aggiunta di un record.

NOTA - Esplorando gli oggetti di questa, come di altri componenti del database Gestione contatti, e portandosi nell’Editor VBA si scoprono molte routine bell’e predisposte. Conviene esaminarle? Sì, ma con cautela da chi è alle prime armi...

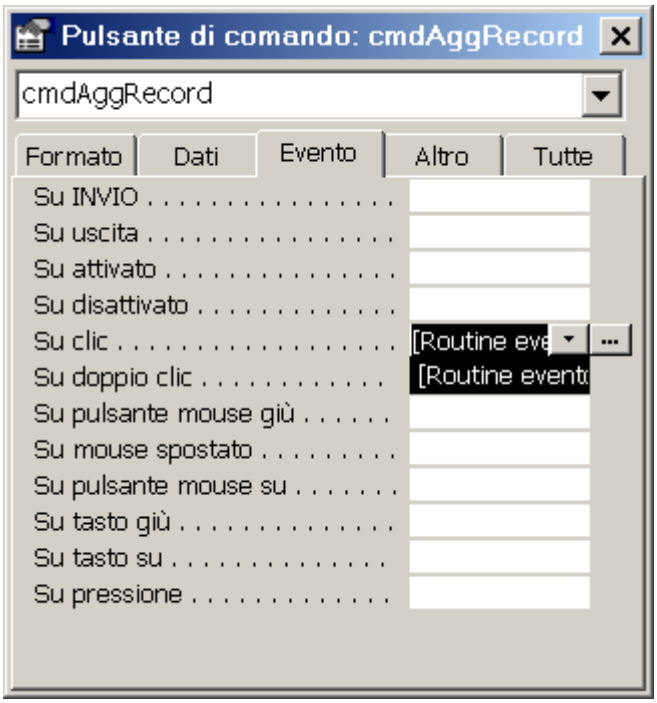
1. Aprire tale maschera in visualizzazione Struttura e nella casella degli strumenti sia attivata l’icona a forma di bacchetta magica, di creazione guidata.
2. Portarsi, agendo sulle scrollbar laterali se occorre, nella zona “Piè di pagina” della maschera, ove sono già presenti diversi pulsanti.
3. Scegliere il controllo Pulsante di comando, posizionarlo nel piè di pagina, accanto alla coppia di pulsanti **1** e **2** e dimensionarlo come al solito. Al termine (a differenza di quanto non accade in Excel VBA) interviene il prima passaggio di creazione guidata, peculiare del controllo scelto.
4. Al primo passaggio scegliere fra i vari comandi chiaramente indicati, suddivisi in categorie, “Operazioni su record / Aggiungi nuovo record”.



5. Nel successivo passaggio scegliere il pulsante di opzione **Testo** e, nella casella accanto, accettare il suggerimento “Aggiungi record”. Sarà l’etichetta stampigliata sul pulsante.
6. Nell’ultimo passaggio sostituire invece il nome suggerito - “Comando 46” o simili - con un più pregnante **cmdAggRecord** (verrà impiegato nelle routine VBA). Concludere con un clic su **Fine**.

Come i più accorti sicuramente si attendono, il wizard ha fatto qualcosa di più che creare un pulsante di comando e relative proprietà, ha creato del codice VB. Lo scopriamo facilmente, oltre che portandosi brutalmente nell’Editor, in un altro modo specifico:

7. Mantenendo attivato il neonato pulsante, portarsi nella sua finestra Proprietà. Nella scheda ***Evento*** noteremo che la casella *su clic* reca la scritta “[routine d’evento]”, che a chiare note denuncia la presenza di una procedura associata al clic sul controllo. Chi mastica VB indovina già che si tratta di una Sub *cmdAggRecord_Click*...



8. E infatti basta dare un clic sui tre puntini accanto per scoprire l'altarino:

```
Private Sub cmdAggRecord_Click()
On Error GoTo Err_cmdAggRecord_Click

    DoCmd.GoToRecord , , acNewRec
Exit_cmdAggRecord_Click:
    Exit Sub

Err_cmdAggRecord_Click:
    MsgBox Err.Description
    Resume Exit_cmdAggRecord_Click

End Sub
```

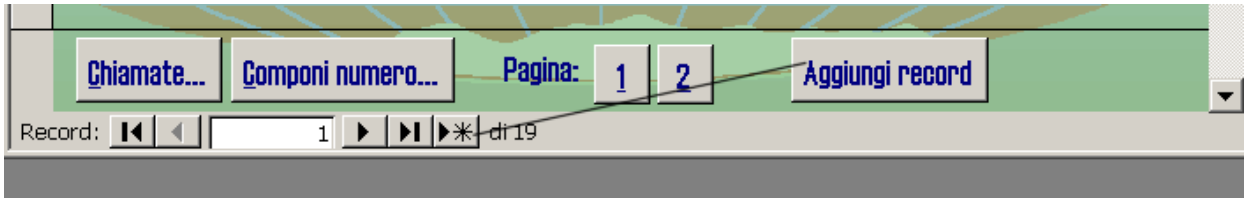
Come si constata, il wizard non solo ha inserito *DoCmd.GoToRecord , , acNewRec*, il comando per l'aggiunta di un nuovo record (ma come avete fatto a indovinare?) ma un canovaccio di istruzioni di gestione errori. Quali errori? Quelli che dovremmo analizzare in funzione di un dato problema... Al momento tale canovaccio è nuovo, comunque è bell'e pronto. Almeno per il momento, non ne abbiamo bisogno, comunque una piccola cosa possiamo aggiungere, la seguente riga da inserire a valle di *DoCmd.GoToRecord...*:

```
Nome.SetFocus
```

A che serve? Ricordiamo che il metodo *SetFocus* di un controllo pone il "focus", ossia focalizza, rende disponibile quel certo controllo. In questo caso *Nome* è il controllo casella di testo del campo Nome della maschera.

Il test della nuova creatura è presto fatto, dopo aver chiuso, e salvato!, la vista Struttura: lancia la maschera Contatti modificata, un clic sul nuovo pulsante ne presenta all'utente una vuota, col campo Nome pronto a essere riempito.

Qualcuno dirà che lo stesso mestiere viene già svolto dall'ultimo pulsantino di navigazione fra i record, quello marcato con *** :**



Verissimo, ma possiamo immaginare che l'utente finale non conosca questa particolarità.

NOTA - In un'applicazione semichiusa per utenti con poca pratica di Access spesso si eliminano del questi strumentini navigatori di cui le maschere sono di norma dotati: basta impostare a "No" la proprietà *Pulsanti spostamento* della maschera.

Ma adesso ci sembra di udire il suono flebile, ma insistente, di un campanello. Indica la fine di questa pur virtuale lezione e non resta che rinviare tutti alla prossima.